

LOGARITHMIC LOOKUP TABLES

[0001] This application claims priority from U.S. Provisional Application Serial No. 60/305,968, entitled "DIGITAL VOLTAGE GAIN AMPLIFIER IMPLEMENTATION IN LOGARITHMIC DOMAIN IN THE ZERO IF ARCHITECTURE," filed July 16, 2001, the content of which is incorporated herein by reference in its entirety. Co-pending and commonly assigned U.S. Application Serial No. _____, entitled "DIGITAL VOLTAGE GAIN AMPLIFIER FOR ZERO IF ARCHITECTURE" filed the same day as this application, and bearing Attorney Docket No. 010419 is also hereby incorporated herein by reference in its entirety.

FIELD

[0002] The invention relates generally to lookup tables, and more particularly logarithmic lookup tables implemented in wireless communication systems.

BACKGROUND

[0003] One common technique used in wireless communication is code division multiple access (CDMA) signal modulation in which multiple communications are simultaneously conducted over a radio-frequency (RF) spectrum. Some example wireless communication devices that have incorporated CDMA technology include cellular radiotelephones, PCMCIA cards incorporated within computers, personal digital assistants (PDAs) equipped with wireless communication capabilities, and the like.

[0004] A conventional architecture for a CDMA receiver includes a radio-frequency (RF) section and an infrared (IF) section. In particular, the received RF signals are typically filtered in the RF section, converted from RF signals to IF signals for further filtering and scaling by a voltage gain amplifier (VGA) in the IF section, and finally converted to baseband signals. The baseband signals are typically passed through an analog-to-digital (A/D) converter to produce digital samples which can be sent to a digital signal processor for tracking and demodulation.

[0005] The Zero infrared frequency (Zero IF) architecture is a more recent architecture used in CDMA wireless communication devices. Unlike other conventional architectures, the Zero IF architecture converts incoming RF signals directly into baseband signals without first converting the RF signals to IF signals. In particular, the Zero IF architecture makes use of a digital VGA that scales the digital samples produced by the A/D converter. In this manner, the Zero IF architecture eliminates the need for various IF components, including an IF mixer, an IF VGA and IF filters.

[0006] In the heterodyne architecture with an IF section, the IF-VGA controlled by an automatic gain control unit (AGC) is responsible for either expanding or compressing the signal such that it fits in the relatively narrow dynamic range of the A/D converter. The A/D converter can then produce small bit-width (typically 4 bits) numbers so that rest of the hardware that performs signal processing can be simplified. In the Zero-IF architecture however, due to the absence of the IF VGA, the A/D converter is typically designed to have much larger dynamic range resulting in large bit-width numbers at the output.

[0007] Although the Zero IF architecture eliminates the need for IF components, the architecture may require more complicated baseband components, primarily due to the relatively large digital signals (typically 18 bits) generated by the A/D converter. Consequently, a digital VGA is implemented at baseband to scale the large bit-width signals from the A/D converter. The Zero IF architecture may implement a relatively wide multiplier (typically an 18-bit by 18-bit multiplier) to scale the large digital signals. In addition, the digital VGA typically includes a relatively large lookup table (LUT) (often exceeding a kilobyte or more) to convert values received from the AGC unit from logarithmic units in decibels (dB) to linear values for controlling the gain of the digital VGA. In operation, for example, the digital VGA multiplies the linear digital signals received from the A/D converter by a linear gain value obtained from the LUT. For these reasons, wireless communication devices incorporating the Zero IF architecture may have significant cost even though the IF components have been eliminated.

[0008] The amount of memory space needed to store one or more lookup tables is generally proportionate to the amount of information stored within the lookup

tables. Thus, as lookup tables become larger, the use of lookup tables can become memory intensive. For some wireless communication devices, memory space can be quite limited, thus making lookup tables difficult and/or costly to implement.

SUMMARY

[0009] In general, the invention is directed toward techniques for efficiently generating logarithmic values from linear input. In particular, the techniques exploit properties of a logarithmic function to reduce the memory requirements needed to implement lookup tables (LUTs). The techniques may utilize non-uniform sampling over a logarithmic or logarithmic-like function to reduce the number of entries within one or more LUTs used when performing such computations. These techniques may be used in a wide-variety of applications, including spread spectrum wireless communication systems.

[0010] In particular, the techniques may involve separating one or more input variables into an exponent component and a mantissa component. For example, the input variables may comprise digital values of a signal, such as a digital baseband signal processed within a wireless communication device. Each of the components of the input variables can be converted from a first domain to a second domain using corresponding LUTs. For example, the exponent component can be converted using an exponent LUT. Similarly, the mantissa component can be converted using a mantissa LUT. After conversion, the components can then be combined for computations and other operations performed within the second domain. The first domain may be a linear domain and the second domain may be a non-linear domain. For example, the second domain may be a logarithmic domain having decibels as units with the desired resolution.

[0011] The size of the mantissa LUT, in particular, can be reduced by storing only a subset of a total number of mantissa values, and then deriving other mantissa values from the subset as needed. For example, according to the principles of the invention, a portion of the mantissa entries, referred to as a segment, can be stored, and mantissa values of the remaining segments can be derived. The other values may be derived, for example, by adjusting an input

address used to select entries from the mantissa LUT, typically by shifting the input address or by interpolating the input address to a new input address. The modified input address can then be used to select a desired value from the mantissa LUT.

5 **[0012]** In one embodiment, the techniques may be carried out according to one or more methods. For example, a number representing a digital value of a signal may be separated into an exponent component and a mantissa component in a first domain. The exponent component can then be converted from the first domain to a second domain, and separately, the mantissa component can be
10 converted from the first domain to the second domain. The exponent component and the mantissa component can then be combined in the second domain, and various computing functions or other operations can be performed within the second domain, such as amplifying the signal within a digital amplifier of a wireless communication device. Notably, for the mantissa component, a subset
15 of a total number mantissa values may be stored in a mantissa LUT. Mantissa values not stored in the mantissa LUT can be derived from the subset.

[0013] The described techniques may be implemented in software, hardware, or a combination thereof. Accordingly, the invention may comprise a computer readable medium carrying program code that when executed performs one or
20 more of the above-mentioned methods.

[0014] In one embodiment, a wireless communication device or other apparatus may include an exponent extractor that extracts an exponent component from a number, and a mantissa extractor that extracts a mantissa component from the number. The apparatus may also include a first LUT used to convert the
25 exponent component from a first domain to a second domain, and a second LUT used to convert the mantissa component from the first domain to the second domain. The apparatus may also include an adder that combines the exponent component and the mantissa component in the second domain. The apparatus may comprise a digital voltage gain amplifier used in a wireless communication
30 device.

[0015] In particular, a wireless communication device according to the invention may include an antenna that receives an RF signal and an RF mixer that generates a baseband signal from the RF signal. In addition, the wireless

communication device may include the voltage gain amplifier described above for scaling the baseband signal. The wireless communication device may also include a digital signal processor that processes at least part of the scaled baseband signal.

5 [0016] The invention can provide a number of advantages. In particular, the invention can significantly reduce memory requirements for computer devices implementing one or more LUTs. A logarithmic conversion unit may utilize different LUTs for the exponent and the mantissa of a floating point number, as described in detail below, which can facilitate the reduction of memory requirements. In particular, a portion of the mantissa entries may be stored in the mantissa LUT, and the additional mantissa values can be generated or approximated as needed. In some cases, the subset of mantissa entries may comprise a non-uniform sample of a total number of mantissa values. These or other non-uniform sampling techniques can be used to drastically reduce memory requirements of a mantissa LUT.

10 [0017] The invention may be implemented in any computer device using LUTs. Although various details of the invention are described in the context of a wireless communication device, the invention is not necessarily limited in that respect. Rather, the techniques can be used with any computer device implementing LUTs to reduce the size of the LUTs.

15 [0018] Additional details of these and other embodiments are set forth in the accompanying drawings and the description below. Other features, objects and advantages will become apparent from the description and drawings, and from the claims.

25 BRIEF DESCRIPTION OF DRAWINGS

[0019] FIG. 1 is a block diagram illustrating a wireless communication device according to the invention.

[0020] FIG. 2 is a flow diagram according to an embodiment of the invention.

30 [0021] FIG. 3 is a block diagram illustrating an embodiment of digital VGA according to the invention.

[0022] FIG. 4 is a block diagram, illustrating in greater detail, one embodiment of digital VGA.

[0023] FIGS 5A-5C illustrate three different exemplary embodiments of an exponential conversion unit according to the invention.

[0024] FIG. 6 is a block diagram, illustrating in greater detail, one implementation of a logarithmic conversion unit.

[0025] FIGS. 7A and 7B are a series of two graphs that further illustrate the concept behind the embodiment of logarithmic conversion unit illustrated in FIG. 6.

[0026] FIGS. 8 and 9 are flow diagrams according to the invention.

DETAILED DESCRIPTION

[0027] FIG. 1 is a block diagram of an exemplary wireless communication device (WCD) 100 according to the invention. Although not so limited, various embodiments of the invention are described in reference to a receiver of wireless communication device that implements the Zero IF architecture, as illustrated in FIG. 1. In that case, WCD 100 converts incoming RF signals directly into baseband signals and, specifically, does not first convert the RF signals to IF signals.

[0028] WCD 100 includes antenna 104 that receives incoming RF signals. For example, the incoming RF signals may comprise code division multiple access (CDMA) modulated signals sent from a CDMA base station. An RF signal received by antenna 104 can be passed through low-noise amplifier (LNA) 108 before being mixed down to baseband by RF mixer 112. For example, RF mixer 112 may receive carrier waveforms produced by frequency synthesizer 116 that utilize a local clock of WCD 100 as a timing reference. The local clock, for instance, may comprise a voltage controlled oscillator 120, such as a voltage controlled temperature compensated crystal oscillator (VCTCXO). As desired, WCD 100 may also include additional components (not shown).

[0029] RF mixer 112 produces baseband signal 113, which can be filtered, e.g., by filter 124, and sampled by analog to digital (A/D) converter 128 to produce corresponding digital values of the signal, e.g., digital baseband signal 129. Digital VGA 132 scales digital baseband signal 129, either by amplifying or attenuating the digital values according to logarithmic gain value 135 received from automatic gain control unit (AGC) 134.

[0030] After scaling by digital VGA 132, the scaled digital baseband signal is provided to rake receivers 136, which separate and track signals received from different sources, e.g., different base stations. For example, rake receivers 136 may include a number of “fingers” that perform despreading, Walsh decoding and accumulation, pilot time and frequency tracking. Each finger outputs pilot and data symbols for the corresponding path to digital signal processor (DSP) 140. DSP 140 then performs symbol demodulation and/or other signal processing.

[0031] As described above, although the Zero IF architecture eliminates the need for IF components, the architecture has typically required conventional devices to incorporate more complicated baseband components. However, unlike a conventional digital VGA that may implement a relatively wide multiplier, digital VGA 132 operates in a logarithmic domain. In particular, digital VGA 132 scales digital baseband signal 129 within the logarithmic domain, and in accordance with the logarithmic gain value 135 received from AGC 134. In this manner, digital VGA 132 does not incorporate a complex and expensive multiplier, but utilizes a more simple and relatively low cost adder. In addition, because digital VGA 132 operates in the logarithmic domain, the gain value 135 received from AGC 134 may be used without conversion to a linear value, thus removing the need for the conventional lookup table (LUT) used for that purpose.

[0032] As discussed in further detail below, digital VGA 132 may utilize a logarithmic conversion unit (not shown in FIG. 1) to convert digital baseband signal 129 from a linear domain to a logarithmic domain, and may utilize an exponential conversion unit (not shown) to convert a scaled baseband signal from the logarithmic domain back to the linear domain. The exponential conversion unit and the logarithmic conversion unit may utilize lookup tables. Additional techniques described below, however, can be used to reduce the size of the lookup tables and thus reduce memory requirements. In other embodiments, the exponential conversion unit and logarithmic conversion unit may utilize algorithms, possibly in combination with smaller sized lookup tables, to dynamically generate the appropriate conversion values.

[0033] FIG. 2 is a flow diagram illustrating a high-level process in which digital VGA 132 (FIG. 1) scales a baseband signal in the logarithmic domain. As

shown, digital VGA 132 converts a linear baseband signal from a linear domain to logarithmic domain (202). As described in detail below, digital VGA 132 may incorporate a logarithmic conversion unit that makes use of lookup tables to perform the conversion. The logarithmic domain may have units of decibels scaled to the desired resolution so that conversion of the gain values can be avoided. For example, a resolution of 2/15 dB may be used in a WCD, corresponding to the defined resolution of the gain.

[0034] Digital VGA 132 adds the baseband signal in the logarithmic domain to gain values received from AGC 134, thereby generating a scaled baseband signal in the logarithmic domain (204). The scaled baseband signal may be, for example, amplified or attenuated depending on the gain values. Digital VGA 132 converts the scaled baseband signal back from the logarithmic domain to the linear domain (206). Digital VGA 132 may incorporate an exponential conversion unit that makes use of lookup tables to perform the conversion. The scaled linear baseband signal can then be processed, for example, by rake receivers 136 and digital signal processor 140 (208).

[0035] FIG. 3 is a block diagram illustrating an example embodiment of digital VGA 132 that operates within the logarithmic domain. As shown, digital VGA 132 includes a logarithmic conversion unit 302 that converts an incoming linear baseband signal 129 into a logarithmic domain, such as by converting the linear digital values to logarithmic values 305 having units of decibels with a desired resolution. In one embodiment, logarithmic conversion unit 302 includes a logarithmic LUT that maps acceptable values for linear baseband signals to the logarithmic values. Such a LUT, however, can require significant memory space. For this reason, various additional techniques are described to reduce the size of the logarithmic LUT, and thereby reduce memory space requirements.

[0036] Digital VGA 132 includes an adder 306 rather than a conventional multiplier. In particular, digital VGA 132 exploits the logarithmic domain, and replaces the multiplication function with addition. In other words, digital VGA 132 exploits the fact that $\text{Log}(X * Y) = \text{Log}(X) + \text{Log}(Y)$, to effectively replace a multiplier with adder 306. WCD 100, however, may further process the amplified baseband signal in the linear domain. Therefore, digital VGA 132 may

also include an exponential conversion unit 310 to convert the baseband signal back into the linear domain after it has been scaled.

[0037] The gain of digital VGA 132 can be controlled by feedback from AGC 134. In particular, AGC 134 may output a gain value 135 in the logarithmic domain, such as in units of decibels having the desired resolution. In this manner, digital VGA 132 need not convert the gain value 135 prior to application by adder 306. Rather, digital VGA 132 may amplify or attenuate linear baseband signal 129 accordingly, simply by adding the gain value 135 to linear baseband signal 129 in the logarithmic domain.

[0038] Exponential conversion unit 310 converts the scaled baseband signal from the logarithmic domain back to the linear domain. For example, the exponential conversion unit 310 may use one or more LUTs, as described in greater detail below. In addition, the techniques described below may be used to significantly reduce the size of the one or more exponential LUTs to reduce memory requirements.

[0039] In one particular case, digital VGA 132 operates in a logarithmic domain having units of decibels scaled to a desired resolution. As mentioned above, operating in a logarithmic domain having units of decibels can be particularly advantageous, because in that case the gain value generated by AGC 134 can be used without conversion. In other embodiments, however, the invention may operate in other logarithmic domains or non-linear domains. In still other embodiments, logarithmic conversion unit 302 and exponential conversion unit 310 implement algorithms in addition to, or as an alternative to, the various LUTs.

[0040] FIG. 4 is a block diagram of one embodiment of digital VGA 132 illustrated in greater detail. In particular, logarithmic conversion unit 302 is illustrated as including a logarithmic lookup table (LOG LUT) 402. In addition, logarithmic conversion unit 302 includes hardware that separates an incoming signed baseband signal into an unsigned absolute value (ABS as shown at 406) and a sign bit (as shown at 410). In other words, the sign bit can be removed from the baseband signal within logarithmic conversion unit 302 and then propagated and reinserted in exponential conversion unit 310. The logarithm of a negative number is undefined. Thus, removing the sign bit from the baseband

signal avoids the scenario where hardware within the digital VGA attempts to operate on negative numbers within the logarithmic domain. Rather, specific hardware within digital VGA 132 may operate only on positive numbers within the logarithmic domain once the sign bit has been removed.

5 **[0041]** In one particular example, A/D converter 128 (FIG. 1) produces the digital baseband signal 129 as an 18-bit binary number, including a sign bit, 7 integer bits, and 10 decimal bits. The sign bit is separated from the baseband signal and the remaining 17-bit binary number, including 7 integer bits and 10 decimal bits, is fed into LOG LUT 402. As a result of the lookup, the 17-bit binary number is converted into a 9-bit signed number, which is fed into adder 306. Adder 306 also receives a 9-bit signed number from AGC 134 representing a gain value in units of decibels with the desired resolution. Notably, the gain value is already in the logarithmic domain, and thus requires no adjustment prior to input into adder 306. Adder 306 adds the respective inputs to produce a 10-bit result, which is fed into exponential conversion unit 310. In one example, a relatively simple and low cost 9-bit adder is used.

10 **[0042]** Exponential conversion unit 310 may include saturation unit 424 and at least one exponential LUT 428. The saturation unit can be used to saturate the baseband signal in the logarithmic domain. For example, continuing with the example above, the 10-bit signed result, which is fed into exponential conversion unit 310, can be saturated to 7-bits by saturation unit 424 to span a dynamic range represented by 128 possible values. The output of saturation unit 424 is fed to exponential LUT 428 along with the sign bit (as shown at 410) to generate a 4-bit signed result representing the scaled baseband signal in the linear domain.

15 Performing saturation prior to the lookup in exponential LUT 430 can save memory space by reducing the required size of individual entries in exponential LUT 430. Moreover, truncation operations can be incorporated into exponential LUT 430. In other words, any conventional truncation operations that would need to follow a conventional multiplier type of VGA can be incorporated into

20 exponential LUT 430 in accordance with the invention simply by defining the appropriate bit-width of the output of exponential LUT 430.

25 **[0043]** FIGS 5A-5C illustrate three different exemplary embodiments of an exponential conversion unit according to the invention. The saturation unit 424

30

illustrated in FIG. 4 is not shown in FIGS. 5A-5C, but could be included for saturation prior to the lookups to reduce the size of one or more LUTs within the exponential conversion unit. In addition, entries in the one or more LUTs within the exponential conversion unit may have output bit-widths that effectively truncate digital input signals.

[0044] The exponential conversion unit 502 illustrated in FIG. 5A, includes different LUTs for positive and negative values. In particular, exponential conversion unit 502 includes EXP_LUT(pos) 506 that stores the 128 possible positive 3-bit values, and also includes EXP_LUT(neg) 510 that stores the 128 possible negative 4-bit values. The sign bit (as shown at 410) is used to provide input signal 514 used by multiplexer 518 to select output from the appropriate LUT. Thus, when the sign bit identifies a negative number, multiplexer 518 selects output from EXP_LUT(neg) 510. When the sign bit identifies a positive number, multiplexer 518 selects output from EXP_LUT(pos) 506.

[0045] FIG. 5B illustrates yet another configuration of an exponential conversion unit. In particular, exponential conversion unit 540, as illustrated in FIG. 5B, uses a single EXP_LUT(pos) 544 and replaces the EXP_LUT(neg) with an adder 548 that is used to generate the complimentary negative value from a positive value read from EXP_LUT(pos) 544. For example, given a positive value X stored in EXP_LUT(pos), the complimentary negative value may be generated by the appropriate complimentary equation, in one case $(-X-1)$. In other cases, however, where the negative and positive values are perfectly symmetric about a Y-axis, the twos-complement equation of $-X+1$, or the like, may be used to generate the negative values from the positive values.

[0046] Exponential conversion unit 540, as illustrated in FIG. 5B, can provide advantages by further reducing memory requirements. In particular, exponential conversion unit 540 utilizes a single 128-bit by 3-bit LUT, and adder 548. Exponential conversion unit 502 as illustrated in FIG. 5A, on the other hand, requires more memory space, but eliminates the additional adder. Each of the embodiments of FIGS. 5A and 5B has advantages and disadvantages, which can be considered in determining the best configuration for a given implementation.

[0047] FIG. 5C illustrates still another configuration of an exponential conversion unit. Like the embodiment of FIG. 5B, exponential conversion unit

560 illustrated in FIG. 5C uses a single EXP_LUT(pos) 544. However, exponential conversion unit 560 generates negative values by inverting bits selected from EXP_LUT(pos) 544 and appending the sign bit. One's complement unit 568 can be used to perform the inversion and append unit 574 can be used to append the sign bit after multiplexer 518 selects the desired output. The embodiment of FIG. 5C avoids the need for an EXP_LUT(neg) as illustrated in FIG. 5A and also avoids the need for an adder as illustrated in FIG. 5B.

[0048] FIG. 6 is a block diagram, illustrating in greater detail, one implementation of logarithmic conversion unit 302. As mentioned above, logarithmic conversion unit 302 is used to convert digital values of a signal, e.g., an incoming linear baseband signal 129, into a logarithmic domain. While logarithmic conversion unit 302 can be realized by implementing a single LUT that maps acceptable values for linear baseband signal to the logarithmic domain, an alternative configuration, as illustrated in FIG. 6, can significantly reduce memory requirements.

[0049] In particular, as illustrated in FIG. 6, logarithmic conversion unit 302 includes various hardware for manipulating a floating point number. For example, an N-Bit positive binary number $X(p)$ can be expressed as $X(p) = m \cdot 2^E$, where E is referred to as the exponent and represents the position of the most significant bit in $X(p)$ that is set to a one, and m is referred to as the mantissa and represents the remaining $E-1$ least significant bits.

[0050] In a logarithmic domain having decibel units, the above floating point equation can be expressed as:

$$20 \cdot \log(X(p) / 2^{10}) \cdot R_{dB} = 20 \cdot \log(2^E / 2^{10}) \cdot R_{dB} + 20 \cdot \log(m) \cdot R_{dB},$$

where R_{dB} is the inverse of the desired decibel resolution. The floating point equation in the logarithmic domain in decibel units can then be expressed as:

$$X = X_E + X_m,$$

where X_E is the decibel value of the exponent and X_m is the decibel value of the mantissa.

[0051] In this particular domain, X_E can be stored as a table of $N=17$ entries, where each entry is 9-bits wide. Furthermore, X_m can be thought of as N segments. However, only one of the segments needs to be stored as a table, and values for all other segments can be derived from entries of the stored segment.

For example, the m^{th} segment includes 2^{m-1} elements, each of which can be derived from an element of the $(m + n)^{\text{th}}$ segment by left-shifting the input address of the $(m + n)^{\text{th}}$ segment by n -bits. In addition, a nearest neighbor interpolation can be performed to derive elements of the $(m + n)^{\text{th}}$ segment from elements of the m^{th} segment. In particular, to interpolate an element of the $(m + n)^{\text{th}}$ segment from an element of the m^{th} segment, the input address of the m^{th} segment can be rounded by n -bits and used as the input address of the $(m + n)^{\text{th}}$ segment.

[0052] Referring again to FIG. 6, logarithmic conversion unit 302 may include an exponent extractor 602 and a mantissa extractor 606. An exponent LUT 612 can be used to generate the logarithmic value of the exponent. In addition, a mantissa LUT 616 can be used to generate the logarithmic value of the mantissa. These values can then be combined by adder 620, and possibly truncated by truncate unit 624.

[0053] In one embodiment, mantissa LUT 616 maps all possible values of the various mantissa segments. However, to reduce the size of mantissa LUT 616 and thereby reduce memory requirements, address manipulation logic 624 can be implemented to allow values of one segment of the mantissa to be generated from a different segment of the mantissa. In this manner, only a single segment of the mantissa can be stored in mantissa LUT 616.

[0054] Address manipulation logic 624 may operate as follows. Assuming that mantissa LUT 616 includes 2^U entries, then

if $E > U$, let input address $M' = \text{round}(M/2^{(E-U)})$, and

if $E \leq U$, let $M' = \ll (U-E)$.

In other words, if $E > U$, then M' is interpolated and defined as the nearest neighbor input address in the stored mantissa LUT 616, and if $E \leq U$, M' is obtained by left shifting the input address by $(U-E)$ bits. M' can then be fed into the stored segment to yield the appropriate converted value in the logarithmic domain.

[0055] FIGS. 7A and 7B are a series of two graphs that further illustrate the concept behind the embodiment of logarithmic conversion unit 302 illustrated in FIG. 6. Again, the floating point equation in the logarithmic domain having

decibel units can be expressed as $X = X_E + X_m$, where X_E is the decibel value of the exponent and X_m is the decibel value of the mantissa. FIG. 7A is a graph of the decibel value of X as a function of linear input, in comparison to a graph of the decibel value of X_E for the same input. As can be seen in FIG. 7A, the value of X_E deviates slightly from the value of X at various different inputs.

[0056] FIG. 7B is a graph of the decibel value of X_m . X_m can be viewed as the variation between X and X_E at the various different inputs. Thus, as shown in FIG. 6, X can be generated from exponent LUT 612 and mantissa LUT 616. In particular, exponential LUT 612 is used to generate the logarithmic value of the exponent, and mantissa LUT 616 is used to generate the logarithmic value of the mantissa. Importantly, however, to save memory space, mantissa LUT 616 need only store entries for one of the spikes (i.e. segments) illustrated in FIG. 7A. The values of the other spikes can then be generated using address manipulation logic 624 as described above.

[0057] The segment (or spike as illustrated in FIG. 7B) that is stored in mantissa LUT 616 can be chosen depending on the level of accuracy required. For example, the full logarithmic mantissa table can be derived without any loss by storing the last segment (in this case the 17th segment). However, that would result in the LUT having $2^{17-1} = 65,536$ entries. Notably, because logarithmic functions tend to flatten at large inputs, a much smaller segment can be chosen without a significant loss in performance in a wireless communication device. In particular, experiments have showed that storing the 7th segment yields acceptable approximations for all inputs. This would require a mantissa LUT having only $2^{7-1} = 64$ entries.

[0058] Choosing a relatively small numbered segment for inclusion in the mantissa table effectively results in non-uniform sampling over the logarithmic function. At lower inputs, e.g., in lower segments, where the logarithmic function changes relatively rapidly, all values in lower numbered segments can be generated from the larger numbered segment. However, at higher inputs, where the logarithmic function changes much more slowly, the values are interpolated from the stored segment. This can be viewed as a sort of non-uniform sampling of the logarithmic function, wherein at lower inputs more data points are preserved, but at higher inputs, fewer data points are preserved.

Experiments have shown that such non-uniform sampling does not result in significant performance reduction in a WCD.

[0059] To account for any unexpected variables, such as quantization noise, the resolution of both exponent LUT 612 and mantissa LUT 616 can be increased by K-bits to help insure that the data is correct and non-corrupted. Quantization noise may be introduced whenever a floating point LOG function is represented by finite bit-width digital numbers. Experiments have showed that increasing the resolution of exponent LUT 612 and mantissa LUT 616 by $K = 3$ bits can be sufficient to compensate for quantization noise in a WCD.

[0060] FIG. 8 is a flow diagram illustrating a process that can be used to reduce memory requirements when implementing one or more LUTs. The process illustrated in FIG. 8 may be particularly effective when used in a WCD, but is not so limited. For example, the process of FIG. 8 may be used in any computer device implementing lookup tables.

[0061] As shown, a number is separated into an exponent component and a mantissa component in a first domain (802). For example, the first domain may be a linear domain and the number may be binary floating point number representing digital values of a signal. In that case, the exponent component can be extracted from the number simply by identifying and storing the most significant bit that is set to one. Similarly, the mantissa component can be extracted from the number simply by storing the remaining least significant bits.

[0062] The exponent component can be converted from the first domain to a second domain, such as a non-linear domain (804). Separately, the mantissa component can be converted from the first domain to the second domain (806).

In one particular case, the second domain is a logarithmic domain having units of decibels with the desired resolution. Once converted to the second domain (804 and 806), the exponent and mantissa components can be combined (808). For example, an adder can be implemented to facilitate the combination simply by summing the exponent and mantissa components. If desired, the combination can then be truncated or rounded.

[0063] The conversion of the exponent component can be done using a first LUT, e.g., an exponent LUT. Similarly, the conversion of the mantissa component can be done using a second LUT, e.g., a mantissa LUT.

Alternatively, either or both of the conversions may be achieved using an algorithm, or an algorithm and a LUT in combination.

[0064] The exponent LUT may be relatively small, and may have entries corresponding to the possible bit positions of the exponent. For example, if the number is a 17-bit number, the exponent LUT may have only 17 entries. The mantissa LUT, on the other hand, may include a much larger number of entries. For example, if the number is a 17-bit number, the mantissa LUT could include an extremely large number of entries. For this reason, additional techniques can be used to further reduce the size of the mantissa LUT, and thus save memory space.

[0065] FIG. 9 is a flow diagram illustrating a process that can be used to significantly reduce the size of the mantissa LUT. As shown, only a subset of a total number of mantissa values are stored in the mantissa table (902). Values not included in the subset can then be derived from the subset (904). In this manner, the memory requirements for implementing a mantissa LUT can be reduced.

[0066] Referring again to FIG. 7B, the subset of mantissa values may correspond to one of the spikes illustrated in FIG. 7B. The values in all of the other spikes can be generated either by shifting the input and selecting an entry in the subset based on the shifted input, or by interpolating input and selecting an entry in the subset based on the interpolated input. If a value in a smaller numbered mantissa segment is needed (i.e., a segment located to the left of the stored segment in FIG. 7B), the shifting operation can be used. If a value in a larger number mantissa segment is needed (i.e., a segment located to the right of the stored segment in FIG. 7B), the interpolation operation can be used. In this manner, the total number of possible mantissa values can be generated, as needed, from a much smaller subset of mantissa entries stored in a table, thus saving memory space.

[0067] In particular, if a value in a larger number mantissa segment is needed, the input address of that value is interpolated and defined as the nearest neighbor input address within the stored mantissa segment. If a value in a smaller numbered mantissa segment is needed, the input address of that value is left shifting by a number of bits corresponding to the difference between the number of the stored segment and the number of the segment for the desired value. The

interpolated address or shifted address can then be fed into the stored segment to yield the appropriate converted value.

[0068] As mentioned above, the particular subset is stored in the mantissa LUT can be chosen depending on the level of accuracy required. Notably, because logarithmic functions tend to flatten at large inputs, a small segment can be used without a significant loss in performance in a wireless communication device. In particular, experiments showed that storing the 7th segment yielded acceptable approximations for all inputs. This would require a mantissa LUT having only $2^{7-1} = 64$ entries.

[0069] In still other embodiments, a somewhat larger mantissa LUT, including entries that comprise non-uniform sampling of the logarithmic or logarithmic like functions can be used. For example, the entries could be pre-computed using the techniques described above. In other words, a segment of the mantissa could be selected and all of the values in smaller segments could be pre-computed and included within the LUT. However, for larger segments, the entries could be pre-computed according to the interpolation technique described above. In this manner, every segment stored within the LUT would have no more entries than the chosen segment, thus limiting the amount of required memory space. Although the mantissa LUT would be larger than a LUT storing only the single segment, the address manipulation logic would not be needed. In designing a particular implementation, the additional memory space required can be weighed against the inclusion of address manipulation logic to determine the best, or most efficient configuration.

[0070] The various LUTs described above may be stored in memory and accessed as needed. In some cases, methods described above may be implemented in software, such as program code. For example, the program code can be loaded into memory and then executed in a processor. The program code can be initially carried on computer-readable media such as a hard drive or magnetic, optical, magneto-optic, phase-change, or other disk or tape media. Alternatively, the program code may be loaded into memory from electronic computer-readable media such as EEPROM, or downloaded over a network connection. If downloaded, the program code may be initially embedded in a carrier wave or otherwise transmitted on an electromagnetic signal. The program

code may be embodied as a feature in an application program providing a wide range of functionality.

[0071] If the invention is implemented in program code, the processor that executes the program code may take the form of a microprocessor and can be integrated with or form part of a PC, Macintosh, computer workstation, hand-held data terminal, palm computer, WCD, wireless base station, network router, or the like. The memory may include random access memory (RAM) storing program code that is accessed and executed by processor to carry out the various method described above.

[0072] Various embodiments of the invention have been described. For example, a digital VGA has been described for use in a WCD. In particular, the digital VGA operates within the logarithmic domain, and replaces a conventional multiplier and a conventional exponential lookup table having a size greater than 1000 bytes, with an adder, logarithmic LUTs of nominal size (i.e. approximately 200 bytes of total space) and a smaller exponential table of nominal size (i.e. approximately 128 bytes). Nevertheless, various modifications may be made without departing from the spirit and scope of the invention. For example, rather than implementing lookup tables, the invention could be realized using algorithms that dynamically convert from logarithmic domains to linear domains, and/or vice versa. In addition, the invention could be implemented in other logarithmic domains. Moreover, the digital VGA according to the invention could be used in other devices, including, for example, a base station within a wireless communication network. For example, any device that implements a multiplier followed by some sort of truncation may benefit from the invention.

[0073] In addition, aspects of the invention could be used in applications other than digital VGAs. For example, any apparatus that performs a multiplication operation followed by a truncation operation could use aspects of the invention to replace a conventional multiplier with an adder. In one case, the invention can be implemented to multiply two baseband signals using an adder rather than a multiplier. In general, the invention can perform multiplication and truncation by converting linear digital values to a logarithmic domain, adding the digital values in the logarithmic domain and then converting the added value back to a linear domain. Moreover, a truncation operation can be incorporated into a lookup

table that is used to convert back to the linear domain simply by choosing output bit-widths to effectively truncate the signals.

5 [0074] Various techniques for reducing the size of LUTs so as to reduce memory requirements have also been described. In particular, many of these techniques have been described in the context of a WCD. However, the invention is not limited in that respect. For example, the techniques used to reduce the size of one or more LUTs could also be used in other computer devices to save memory space. Accordingly, these and other embodiments are within the scope of the following claims.